Homework 1

Angelis Marios-Kasidakis Theodoros

Asynchronous request-reply model with semantics at most once and flexible service

Compile and run instructions :

Compile server : gcc -Wall -g server_1.c -o server_1 -lpthread

Compile client : gcc -Wall -g client1.c -o client1 -lpthread

Run server : ./server_1 20 239.255.255.250 6000

Run client : ./client_1 20 239.255.255.250 6000

pattern : ./executable svcid udp_address udp_port

Client Application: Each customer sends a number for primality test using the sendRequest function provided by the middleware. It gets the answer for this number by using the getReply function, which is also provided by the middleware.

Server Application: Each server checks whether a number is prime or not. The server gets a number using the getRequest function provided by the middleware. It calculates if the number is prime and then sends the result of the calculation using the sendReply function provided by middleware.

STAGES :

Register stage:

When a server is started, it explicitly declares through the register function that it serves a particular service. For this purpose, the middleware creates a thread which is responsible for storing the requests of the clients send to this server.

Discovery Stage:

Each time the client-application sends a request, middleware is responsible to send this request to the server. The middleware finds which servers are available and serve the same type of service.After that,it sends the request to the one with the lowest capacity(load balancing).When the middleware receives ACK, it returns to the application a unique identifier for this request.

Reply Stage:

Every time a client executes sendRequest,apart from all the others,middleware creates a thread to receive the replies sent by the server via sendReply.When the client application executes getReply,if there is a reply itis returned.The client application prints a suitable result message.

Server Failure: In the event that a server fails, the middleware creates a resend thread in order to ensure that the client's request hasn't been lost. Each request has a resend thread that after sleeping RETRANSMISSION_TIME ,resends the request.This will be done MAX_RETRANSMIT times if there is not a response.